SPECIFICATION

PADDING APPLICATION METHOD ENSURING SECURITY OF CRYPTOSYSTEM AND ENCRYPTOR/ DECRYPTOR

TECHNICAL FIELD

[0001] The present invention relates to an encryption / decryption system, and more particularly, to a padding application method and an encryptor/ decryptor for ensuring the security against chosen ciphertext attacks.

BACKGROUND ART

[0002] The social application of cryptographic technology has been vigorously promoted to ensure the security of communication, and effective cryptographic computation is required more than ever. In such trend, the NTRU cryptosystem proposed in "NTRU: A Ring-Based Public Key Cryptosystem" (Non Patent Document 1) written by Jeffery Hoffstein, Jill Pipher and Joseph H. Silverman attracts attention as a high-speed encryption/ decryption computation system with lower memory requirement as compared to the conventional RSA or ElGamal cryptosystem.

[0003] (NTRU Cryptosystem)

The NTRU cryptosystem is a public key cryptosystem as follows.

[0004] First, a key is created in the following manner. Three integers p, q and N are used as public and domain parameters. Besides, the ring $R = Z[X]/(X^N - 1)$ is used. Hereinafter, L(a, b) indicates the total set (a subset of R) of an element $u \in R$ having a coefficients equal to 1, b coefficients equal to -1 and the rest 0 for each degree thereof. Parameters df, dg and d are chosen to set $L_f = L(df, df + 1)$, $L_g = L(dg, dg + 1)$, and $L_\varphi = L(d, d)$. Two polynomials $f \in L_f$ and $g \in L_g$ are randomly selected such that $h = f^{-1}$

g mod q. Then, the private or secret key is the polynomial f, g, while the public key is the polynomial h.

[0005] When the keys have been created, an element m of the subset L_m of R, $m \in L_m$, is encrypted. A polynomial $r \in L_r$ is randomly selected to compute $e = phr + m \bmod q$. Thus, e is output as a ciphertext.

[0006] In order to decrypt the ciphertext e to the original plaintext or cleartext m, $fe = pgr + fm \bmod q$ is computed. Since f, g, r, and m are elements of the subsets L_f, L_g, L_r, and L_m, respectively, $fe = pgr + fm$. Accordingly, $fe \pmod p = m \pmod p$ can be computed. Also, since the m is an element of the subset L_m, $m = m \pmod p$, and therefore, m can be retrieved.

[0007] However, as pointed out in "Lattice Attacks on NTRU", Eurocrypt '97 Springer Lecture Notes in Computer Science, 1997 (Non Patent Document 2) written by Don Coppersmith and Adi Shamir, there are known a various sorts of attacks against the NTRU cryptosystem. As such, some schemes have been proposed to prevent the attacks, in which some kind of padding is applied to a plaintext before NTRU encryption.

[0008] (OAEP + Padding System)

As a padding scheme to secure cryptosystems, for example, the one called OAEP + is known. The OAEP + padding was proposed in "OAEP Reconsidered", Journal of Cryptology 15 (4) (Non Patent Document 3) written by Victor Shoup. The OAEP + padding is a padding scheme as follows.

[0009] First, integers k, k_0, and k_1 are selected as parameters so as to satisfy $k_0 + k_1 \leqq k \leqq L$, where L is the number of elements in the plaintext space.

[0010] Then, $n = k - k_0 - k_1$ is set.

Let G denote a hash function to map a k-bit string to an n-bit string.

Let H' be a hash function to map an $n + k_0$-bit string to a k_1-bit

string.

Let H be a hash function to map an $n + k_1$-bit string to a k_0-bit string.

[0011] Upon receipt of an n-bit plaintext M, a padder randomly selects a k_0-bit string R. Subsequently, the padder computes the exclusive OR s^0 of G(R) and each bit of M as well as $s^1 = H'(R \| M)$ such that $s = s^0 \| s^1$. Incidentally, the symbol " $\|$ " is used to denote concatenation of bit strings. If t denotes the exclusive OR of H(s) and each bit of R, then $w = s \| t$. This w is called "OAEP + padding using the random number R of the plaintext M". The OAEP + padding w thus obtained is encrypted (by a cryptosystem not using random numbers), and a ciphertext e is transmitted to a receiver.

[0012] The receiver decrypts the ciphertext e to obtain w. After decrypting w, a depadder recovers the plaintext M in the following manner. First, through the use of $w = s \| t = s_0 \| s_1 \| t$, the depadder recovers s_0, s_1 and t. Then, the depadder computes the exclusive OR of H(s) and each bit of t to recover R. Also, the depadder computes the exclusive OR of G(R) and each bit of s_0 to recover M. If $s_1 = H'(R \| M)$ is satisfied, the depadder outputs M. Otherwise, the depadder rejects the ciphertext e as invalid and outputs $\perp$.

[0013] The OAEP + padding, however, is a padding scheme proposed to be applied to cryptosystems not using random numbers the computation of an encryption function. Consequently, if the OAEP + padding is applied to a cryptosystem using random numbers such as NTRU, security is not always ensured. In addition, when the OAEP + padding is applied to a cryptosystem using random numbers such as NTRU, not a unique but various application methods may by utilized. Therefore, there is also a problem in that it is not possible to immediately distinguish between secure and insecure padding application methods.

[0014] As just described, the OAEP + padding ensures the security only

for cryptosystems not using random numbers. Besides, there have been some OAEP + or OAEP + like padding schemes introduced for use with the NTRU cryptosystem using random numbers to ensure the security. Reference may be had to, for example, the following documents:

[0015] • Joseph H. Silverman, "Plaintext Awareness and the NTRU PKCS", Technical Report #7 version 2, NTRU Cryptosystems, 1998 (Non Patent Document 4)

• Jeffery Hoffstein and Joseph H. Silverman, "Optimizations for NTRU", Public-key Cryptography and Computational Number Theory (Non Patent Document 5)

• Jeffery Hoffstein and Joseph H. Silverman, "Protecting NTRU Against Chosen Ciphertext and Reaction Attacks", Technical Report #16 version 1, NTRU Cryptosystems, 2000 (Non Patent Document 6)

• Phong Q. Nguyen and David Pointcheval, "Analysis and Improvements of NTRU Encryption Paddings", Crypto 2002 Springer Lecture Notes in Computer Science, 2002 (Non Patent Document 7)

[0016] With all of these padding schemes, attacks against the NTRU cryptosystem has succeeded. The padded version of the NTRU cryptosystem described in Non Patent Document 4 is broken by the algorithm proposed in "A Chosen-Ciphertext Attack against NTRU", Crypto 2000 Springer Lecture Notes in Computer Science, 2000 (Non Patent Document 8) written by Eliane Jaulmes and Antoine Joux.

[0017] Further, the padded version of the NTRU cryptosystem described in Non Patent Documents 5 and 6 is broken by the algorithm proposed in Non Patent Document 7. The padding scheme proposed in Non Patent Document 7 does not protect against attacks presented in "Imperfect Decryption and an Attack on the NTRU Encryption Scheme" (Non Patent Document 9) written by John A. Proos.

[0018] There have been proposed padding schemes other than the OAEP + padding scheme aimed at ensuring the security of cryptosystems using

random numbers such as NTRU. However, each padding scheme has a different disadvantage, and the OAEP + padding is still significant to ensure the security of cryptosystems.

Non Patent Document 1: Jeffery Hoffstein, Jill Pipher and Joseph H. Silverman, "NTRU: A Ring-Based Public Key Cryptosystem"

Non Patent Document 2: "Lattice Attacks on NTRU", Eurocrypt '97 Springer Lecture Notes in Computer Sciences, 1997

Non Patent Document 3: "OAEP Reconsidered", Journal of Cryptology 15 (4)

Non Patent Document 4: "Plaintext Awareness and the NTRU PKCS", Technical Report #7 version 2, NTRU Cryptosystems, 1998

Non Patent Document 5: Jeffery Hoffstein and Joseph H. Silverman, "Optimizations for NTRU", Public-key Cryptography and Computational Number Theory

Non Patent Document 6: Protecting NTRU Against Chosen Ciphertext and Reaction Attacks", Technical Report #16 version 1, NTRU Cryptosystems, 2000

Non Patent Document 7: "Analysis and Improvements of NTRU Encryption Paddings", Crypto 2002 Springer Lecture Notes in Computer Sciences, 2002

Non Patent Document 8: "A Chosen-Ciphertext Attack against NTRU", Crypto 2000 Springer Lecture Notes in Computer Sciences, 2000

Non Patent Document 9: John A. Proos, "Imperfect Decryption and an Attack on the NTRU Encryption Scheme"

DISCLOSURE OF THE INVENTION

PROBLEMS THAT THE INVENTION IS TO SOLVE

[0019] As is described above, with the conventional padded version of NTRU cryptosystems, secure cryptographic communication cannot be

performed.

[0020] It is therefore an object of the present invention to provide a padding application method and an encryptor/ decryptor for achieving secure cryptographic communication by applying appropriate padding to a cryptosystem such as NTRU.

MEANS OF SOLVING THE PROBLEMS

[0021] The present inventor noted that random numbers used for encryption can be retrieved in NTRU cryptosystems, and invented a new padding application method capable of ensuring the security of cryptographic communication.

[0022] (Reason for the Realization)

The reason for the realization of the present invention will be schematically described below. As typical cryptosystems using random numbers, the ElGamal and Paillier cryptosystems are known. In these cryptosystems, a receiver of a ciphertext can recover the original plaintext, but cannot recover the random number. As such, the plaintext and random number need to be handled as completely different data.

[0023] Additionally, in many cryptosystems such as ElGamal and Paillier, the encryption function is a stochastic function, and the domain of the encryption function corresponds to the space of the entire plaintext M, while the range thereof corresponds to the space of the entire ciphertext. Meanwhile, the decryption function is a nonstochastic function, and the domain of the decryption function corresponds to the space of the entire ciphertext, while the range thereof corresponds to the space of the entire plaintext. Again, in these cryptosystems, a random number used for encryption cannot be recovered.

[0024] On the other hand, the NTRU cryptosystem is characterized in that a random number r can be obtained by r = (fe - fm)/ pg after the recovery of a plaintext m. Consequently, in the NTRU cryptosystem, the

plaintext is not necessarily distinguished from the random number differently from such cryptosystems as ElGamal and Paillier.

[0025] Accordingly, in the NTRU cryptosystem, it can be considered that:

[0026] • the encryption function is a function not based on a random number, and the domain of the encryption function corresponds to the space of the entire concatenation of a plaintext M with a random number R, while the range thereof corresponds to the space of the entire ciphertext; and that

[0027] • the decryption function is also a function not based on a random number, and the domain of the decryption function corresponds to the space of the entire ciphertext, while the range thereof corresponds to the space of the entire concatenation of the plaintext M with the random number R.

[0028] As has already been described, the OAEP + padding ensures the security if an encryption function is not based on a random number. Therefore, the OAEP + padding with security assurance is applicable to the data structure of the NTRU cryptosystem in which a plaintext is not distinguished from a random number.

[0029] (Summary of the Invention)

Based on these understandings, the present invention was made. That is, in accordance with the present invention, there is provided a padding application method to be applied to a cryptosystem $E^r(m)$ that allows a ciphertext receiver to recover the value of a random number used by a ciphertext creator to create a ciphertext. With the application of the present invention to such a cryptosystem, security is ensured.

[0030] In accordance with an aspect of the present invention, there is provided a padding application method for applying a padding scheme that ensures the security of cryptosystems not using random numbers to cryptosystems in which the value of a random number used to create a ciphertext can be recovered at the receiving end. The padding application

method comprises the steps of converting input information into a bit string with a prescribed length or less according to the padding scheme, converting the bit string into a first bit string and a second bit string based on a prescribed conversion rule, and supplying an encryption function with the first bit string as data input and the second bit string as random number input. The prescribed conversion rule is a map to map the bit string having a prescribed length or less to the element of the direct product of the set of the first bit strings and the set of the second bit strings, and satisfies the following conditions: the map is injective; the map and inverse map thereof can be computed by a polynomial time; and the encryption function whose domain is the direct product is a one-way function.

[0031] In accordance with another aspect of the present invention, the conversion rule is a rule to divide the bit string into two parts in such a manner as to set the first half of the bit string as the first bit string and the second half of the bit string as the second bit string.

[0032] In accordance with yet another aspect of the present invention, the OAEP + padding is employed as the padding scheme, and the NTRU cryptosystem is employed as the cryptosystem using random numbers.

[0033] Fig. 1 (A) is a conceptual block diagram showing an encryptor adopting a padding application method according to the present invention. Fig. 1 (B) is a conceptual block diagram showing a conventional encryptor adopting an OAEP-based padding scheme. In Figs 1 (A) and 1 (B), the most common parameters are employed; $n = k_0 = k_1$, and r is the bit number of m.

[0034] Referring to Fig. 1 (A), the padding application method of the present invention will be described. First, parameters are selected in the same manner as in the OAEP + padding. More specifically, integers k, k_0, and k_1 are selected as parameters so as to satisfy k_0, + $k_1 < k < L$, where L is the number of elements in the plaintext space. Then, $n = k - k_0 - k_1$ is set. Also, let G denote a hash function to map a k-bit string to an n-bit string, let

H' be a hash function to map an $n + k_0$-bit string to a k_1-bit string, and let H be a hash function to map an $n + k_1$-bit string to a k_0-bit string.

[0035] When an n-bit plaintext M is received, the OAEP + padding is applied thereto. That is, a k_0-bit string R is randomly selected. Subsequently, the exclusive OR s^0 of G(R) and each bit of M as well as s^1 = H'(R ‖ M) are computed such that $s = s^0 \| s^1$. Incidentally, the symbol " ‖ " is used to denote concatenation of bit strings. If t denotes the exclusive OR of H(s) and each bit of R, then $w = s \| t$.

[0036] Next, according to a rule A (hereinafter referred to as a conversion function A) that satisfies the conditions as will be presently described, two bit strings m and r are generated from w. The conversion function A is a map to map a bit string consisting of k bits or less to the element of $L_m \times L_r$, where L_m is the scope of m and L_r is the scope of r. The conversion function A should satisfy the following conditions:

[0037] (1) A is injective;

(2) A and the inverse map thereof can be computed by a polynomial time; and

(3) if an encryption function is denoted by E(m, r), a map E: $A(X) \rightarrow L_e$ is a one-way function (where X is the scope of (m, r) and L_e is the space of the entire ciphertext).

[0038] In the case of the NTRU cryptosystem, for example, a bit string w is divided into a first half bit string and a second half bit string, and they are set as m and r, respectively. When the bit string w has been converted into two bit strings, $e = E^r(m)$ is computed to be encrypted. Thus, a ciphertext e is transmitted to a receiver.

[0039] Having received the ciphertext e, the receiver decrypts it to obtain m. As is described above, according to a characteristic of $E^r(m)$, r can be recovered. Hence, r is recovered. Subsequently, w is recovered based on $w = m \| r$. After the recovery of w, the plaintext M is recovered in the same manner as in the OAEP + padding. More specifically, through

the use of $w = s \| t = s_0 \| s_1 \| t$, s_0, s_1 and t are recovered. Then, the exclusive OR of H(s) and each bit of t is computed to recover R. Also, the exclusive OR of G(R) and each bit of s_0 is computed to recover M. If $s_1 = H'(R \| M)$ is satisfied, M is output. Otherwise, the ciphertext e is rejected as invalid and $\perp$ is output.

EFFECT OF THE INVENTION

[0040] In the following, on the basis of comparison between the present invention and the conventional techniques, the effect of the present invention will be described. As can be seen in Fig. 1 (B) showing an example of a conventional encryptor, in the aforementioned techniques proposed in Non Patent Documents 4 to 7, only m is obtained from a plaintext M according to the OAEP + (or another) padding, and r is obtained in some way or other. Such padding schemes can be applied to cryptosystems such as ElGamal and Paillier in which a plaintext only but no random number can be recovered at the receiving end. However, these padding schemes are ad hoc, and cannot ensure the security. Especially, in the case of NTRU, perfect decryption is possible.

[0041] On the other hand, in accordance with the present invention, a padding scheme is applied to cryptosystems such as NTRU in which the receiver of a ciphertext can recover a random number as well as a plaintext. Differently from the conventional techniques, both m and r are obtained from a plaintext M according to the OAEP + padding and a prescribed conversion rule (function A). In addition, differently from the conventional techniques, when applied to cryptosystems such as NTRU in which the receiver of a ciphertext can recover a random number as well as a plaintext, the padding scheme can ensure the security.

[0042] In other words, in accordance with the present invention, secure cryptographic communication can be performed making use of the NTRU cryptosystem: the high-speed encryption/ decryption computation system

with lower memory requirement.

BEST MODE FOR CARRYING OUT THE INVENTION

[0043] Fig. 2 is a block diagram showing an example of a cryptographic communication system including an encryptor/ decryptor according to the present invention. In Fig. 2, cryptographic communication is performed between communication terminals via a network.

[0044] A sending communication terminal comprises a program control processor 10, a random number generator 11, a program memory 12, a memory 13 and a transmitter/ receiver 14. As will be described later, the sending communication terminal executes programs stored in the program memory 12, necessary for such operations as OAEP + conversion or transformation, conversion based on a conversion function A and NTRU encryption, to encrypt a plaintext or cleartext. Thereby, the sending communication terminal transmits the ciphertext from the transmitter/ receiver 14 to a destination communication terminal via a network. Incidentally, the memory 13 stores information necessary for encryption such as public information and a private or secret key.

[0045] Similarly, a receiving communication terminal comprises a program control processor 20, a random number generator 21, a program memory 22, a memory 23 and a transmitter/ receiver 24. As will be described later, the receiving communication terminal executes programs stored in the program memory 22, necessary for such operations as NTRU decryption, random number recovery, inversion or inverse transformation and OAEP + inversion or inverse OAEP + transformation, to decrypt a ciphertext received through the transmitter/ receiver 24. Thereby, the receiving communication terminal retrieves the plaintext. Incidentally, the memory 23 stores information necessary for decryption such as public information and a private key.

[0046] 1. First Embodiment

Fig. 3 is a functional block diagram showing the construction of an encryptor/ decryptor according to the first embodiment of the present invention. The encryptor/ decryptor of this embodiment comprises an encryptor 100 for encrypting a plaintext, a decryptor 200 for decrypting a ciphertext to a plaintext, a key generator 300, a public information storage 301 for storing public information necessary for encryption/ decryption generated by the key generator 300, and a private key storage 302 for storing private key information necessary for decryption.

[0047] A plaintext is fed to the encryptor 100 through a plaintext input unit 101. The encryptor 100 includes an OAEP + converter 102, a conversion function A-based converter 103 and an NTRU encryption section 104. A ciphertext created by the encryptor 100 is output to, for example, a receiving terminal through a ciphertext output unit 105.

[0048] A ciphertext is fed to the decryptor 200 through a ciphertext input unit 201. The decryptor 200 includes an NTRU decryption section 202, a random number recovery section 203, a conversion function A-based inverter 204 and an OAEP + inverter 205. A plaintext retrieved by the decryptor 200 is output through a plaintext output unit 206.

[0049] 1.1) Key Creation

First, the key creation process will be described.

[0050] Fig. 4 is a flowchart showing the key creation process according to the first embodiment. The key generator 300 selects integers p, q and N which are used as public and domain parameters. As in the NTRU cryptosystem described above, the ring $R = Z[X]/ (X^N - 1)$ is used. L(a, b) indicates the total set (a subset of R) of an element $u \in R$ having a coefficients equal to 1, b coefficients equal to -1 and the rest 0 for each degree thereof. Additionally, the subsets of R are denoted by L_f, L_g, L_r, and L_m (step S11).

[0051] After that, as previously described for the OAEP + padding scheme, the key generator 300 selects integers k, k_0, and k_1 as parameters

such that they satisfy k_0, + $k_1 \leqq k \leqq L$, where L is the number of elements in $L_m \times L_r$. Then, $n = k - k_0 - k_1$ is set.

Let G denote a hash function to map a k-bit string to an n-bit string.

Let H' be a hash function to map an $n + k_0$-bit string to a k_1-bit string.

Let H be a hash function to map an $n + k_1$-bit string to a k_0-bit string (step S12).

[0052] In addition, the key generator 300 determines a conversion function A (step S13). The conversion function A is a map to map a bit string consisting of k bits or less to the element of $L_m \times L_r$. The conversion function A should satisfy the following conditions:

[0053] (1) A is injective;

(2) A and the inverse map thereof can be computed by a polynomial time; and

(3) if an encryption function is denoted by E(m, r), a map E: $A(X) \rightarrow L_e$ is a one-way function (where X is the scope of (m, r) and L_e is the space of the entire ciphertext).

[0054] The key generator 300 creates a key in the same manner as in NTRU. More specifically, two polynomials $f \in L_f$ and $g \in L_g$ are randomly selected such that $h = f^{-1} g \bmod q$. Then, the private key is the polynomial f, g, while the public key is the polynomial h (step S14). The key generator 300 secretly stores the private key f, g in the private key storage 302 (step S15), and also stores the NTRU public key, the hash functions and the conversion function (p, q, N, L_f, L_g, L_r, L_m, k, k_0, k_1, G, H', H, A, h) in the public information storage 301 to open them to public (step S16).

[0055] 1.2) Encryption

Next, the encryption process will be described.

[0056] Fig. 5 is a flowchart showing the encryption process according to

the first embodiment. Referring to Fig. 5, the encryptor 100 receives an n-bit plaintext M through the plaintext input unit 101 (step S21). Then, the encryptor 100 receives public information p, q, N, L_f, L_g, L_r, L_m, k, k_0, k_1, G, H', H, A, and h from the public information storage 301 (step S22).

[0057] Subsequently, the encryptor 100 randomly selects a k_0-bit string R (step S23). The OAEP + converter 102 computes the exclusive OR s^0 of G(R) and each bit of M as well as $s^1 = H'(R \| M)$ such that $s = s^0 \| s^1$. Further, t is set as the exclusive OR of H(s) and each bit of R to have $w = s \| t$. Thereby, the OAEP + converter 102 applies the OAEP + padding to the plaintext (step S24).

[0058] According to (m, r) = A(w), the converter 103 converts a bit string w into two bit strings m and r using the conversion function A (step S25). Here, w is equally divided into a first half bit string m and a second half bit string r. After that, the NTRU encryption section 104 computes e = phr + m mod q to perform NTRU encryption (step S26). Thus, the NTRU encryption section 104 creates a ciphertext and outputs it through the ciphertext output unit 105 (step S27).

[0059] 1.3) Decryption

Lastly, the decryption process will be described.

[0060] Fig. 6 is a flowchart showing the decryption process according to the first embodiment. Referring to Fig. 6, the decryptor 200 receives a ciphertext e through the ciphertext input unit 201 (step S31). Then, the decryptor 200 receives a private key corresponding to the ciphertext from the private key storage 302, and also receives public information corresponding to the private key from the public information storage 301 (step S32).

[0061] The NTRU decryption section 202 decrypts the ciphertext e using the private key and the public information in the same manner as in NTRU. More specifically, the NTRU decryption section 202 computes fe = pgr + fm mod q. Since f, g, r, and m are elements of the subsets L_f, L_g,

L_r, and L_m, respectively, fe = pgr + fm. Accordingly, the NTRU decryption section 202 can computes fe (mod p) = m (mod p). Also, since the m is an element of the subset L_m, m = m (mod p), and therefore, the NTRU decryption section 202 can retrieve m (step S33).

[0062] In addition, based on fe = pgr + fm, the random number recovery section 203 computes r = (fe - fm)/ pg to recover a random number r (step S34).

[0063] Based on $A^{-1}(m, r)$, the inverter 204 recovers $w = s \| t = s_0 \| s_1 \| t$ using the inverse of the conversion function A (step S35). Subsequently, the OAEP + inverter 205 computes the exclusive OR of H(s) and each bit of t to recover R. Also, the OAEP + inverter 205 computes the exclusive OR of G(R) and each bit of s_0 to recover M (step S36).

[0064] Thereafter, the OAEP + inverter 205 verifies the validity of the padding according to whether $s_1 = H'(R \| M)$ is satisfied or not (step S37). If the padding is valid, the OAEP + inverter 205 outputs the plaintext M (step S38). Otherwise, the OAEP + inverter 205 rejects the ciphertext e as invalid and outputs $\perp$ (step S39).

[0065] 2. Second Embodiment

Fig. 7 is a functional block diagram showing the construction of an encryptor/ decryptor according to the second embodiment of the present invention. The encryptor/ decryptor of this embodiment comprises an encryptor 400 for encrypting a plaintext, a decryptor 500 for decrypting a ciphertext to a plaintext, a key generator 300, a public information storage 301 for storing public information necessary for encryption/ decryption generated by the key generator 300, and a private key storage 302 for storing private key information necessary for decryption.

[0066] A plaintext is fed to the encryptor 400 through a plaintext input unit 101. The encryptor 400 includes a random number generator 401, a private key encryption section 402, an OAEP + converter 403, a conversion function A-based converter 404 and an NTRU encryption section 405. A

ciphertext created by the encryptor 400 is output to, for example, a receiving terminal through a ciphertext output unit 105.

[0067] A ciphertext is fed to the decryptor 500 through a ciphertext input unit 201. The decryptor 500 includes an NTRU decryption section 501, a random number recovery section 502, a conversion function A-based inverter 503, an OAEP + inverter 504 and a private key decryption section 505. A plaintext retrieved by the decryptor 500 is output through a plaintext output section 206.

[0068] 2.1) Key Creation

First, the key creation process will be described.

[0069] Fig. 8 is a flowchart showing the key creation process according to the second embodiment. The key generator 300 selects integers p, q and N which are used as public and domain parameters. As in the NTRU cryptosystem described above, the ring $R = Z[X]/ (X^N - 1)$ is used. L(a, b) indicates the total set (a subset of R) of an element $u \in R$ having a coefficients equal to 1, b coefficients equal to -1 and the rest 0 for each degree thereof. Additionally, the subsets of R are denoted by L_f, L_g, L_r, and L_m (step S41).

[0070] After that, as previously described for the OAEP + padding scheme, the key generator 300 selects integers k, k_0, and k_1 as parameters such that they satisfy $k_0, + k_1 \leqq k \leqq L$, where L is the number of elements in $L_m \times L_r$. Then, $n = k - k_0 - k_1$ is set.

Let G denote a hash function to map a k-bit string to an n-bit string.

Let H' be a hash function to map an $n + k_0$-bit string to a k_1-bit string.

Let H be a hash function to map an $n + k_1$-bit string to a k_0-bit string (step S42).

[0071] In addition, the key generator 300 determines a conversion function A (step S43). As previously described, the conversion function A

is a map to map a bit string consisting of k bits or less to the element of $L_m \times L_r$. The conversion function A should satisfy the following conditions:

[0072] (1) A is injective;

(2) A and the inverse map thereof can be computed by a polynomial time; and

(3) if an encryption function is denoted by E(m, r), a map E: $A(X) \rightarrow L_e$ is a one-way function (where X is the scope of (m, r) and L_e is the space of the entire ciphertext).

[0073] The key generator 300 creates a key in the same manner as in NTRU. More specifically, two polynomials $f \in L_f$ and $g \in L_g$ are randomly selected such that $h = f^{-1} g \bmod q$. Then, the private key is the polynomial f, g, while the public key is the polynomial h. The key generator 300 secretly stores the private key f, g in the private key storage 302 (step S44).

[0074] Further, the key generator 300 determines a common or shared key cryptosystem E to be used (step S45), and stores the NTRU public key, the hash functions and the conversion function (p, q, N, L_f, L_g, L_r, L_m, k, k_0, k_1, G, H', H, A, h) in the public information storage 301 to open them to public (step S46).

[0075] 2.2) Encryption

Next, the encryption process will be described.

[0076] Fig. 9 is a flowchart showing the encryption process according to the second embodiment. Referring to Fig. 9, the encryptor 400 receives an n-bit plaintext X through the plaintext input unit 101 (step S51). Then, the encryptor 400 receives public information p, q, N, L_f, L_g, L_r, L_m, k, k_0, k_1, G, H', H, A, and h from the public information storage 301 (step S52).

[0077] Subsequently, the private key encryption section 402 randomly selects a n-bit string M through the random number generator 401 (step S53). Then, the private key encryption section 402 computes $Y = E_M(X)$ to perform shared key encryption (step S54). Incidentally, $E_M(X)$ is obtained

by encrypting the plaintext X according to the shared key cryptosystem E with M as a key.

[0078] After that, the encryptor 400 randomly selects a k_0-bit string R (step S55). The OAEP + converter 403 computes the exclusive OR s^0 of G(R) and each bit of M as well as $s^1 = H'(R \| M)$ such that $s = s^0 \| s^1$. Further, t is set as the exclusive OR of H(s) and each bit of R to have $w = s \| t$. Thereby, the OAEP + converter 403 applies the OAEP + padding to the plaintext (step S56).

[0079] According to (m, r) = A(w), the converter 404 converts a bit string w into two bit strings m and r using the conversion function A (step S57). Here, w is divided into two parts, and a first half bit string is set as m and a second half bit string is set as r. After that, the NTRU encryption section 405 computes e = phr + m mod q to perform NTRU encryption (step S58). Thus, the NTRU encryption section 405 creates a ciphertext e, and outputs it and a ciphertext Y obtained by the shared key encryption as a ciphertext (e, Y) through the ciphertext output unit 105 (step S59).

[0080] 2.3) Decryption

Lastly, the decryption process will be described.

[0081] Fig. 10 is a flowchart showing decryption process according to the second embodiment. Referring to Fig. 10, the decryptor 500 receives a ciphertext (e, Y) through the ciphertext input unit 201 (step S61). Then, the decryptor 500 receives a private key corresponding to the ciphertext from the private key storage 302, and also receives public information corresponding to the private key from the public information storage 301 (step S62).

[0082] The NTRU decryption section 501 decrypts the ciphertext e using the private key and the public information in the same manner as in NTRU. More specifically, the NTRU decryption section 501 computes fe = pgr + fm mod q. Since f, g, r, and m are elements of the subsets L_f, L_g, L_r, and L_m, respectively, fe = pgr + fm. Accordingly, the NTRU

decryption section 501 can computes fe (mod p) = m (mod p). Also, since the m is an element of the subset L_m, m = m (mod p), and therefore, the NTRU decryption section 501 can retrieve m (step S63).

[0083] In addition, based on fe = pgr + fm, the random number recovery section 502 computes r = (fe - fm)/ pg to recover a random number r (step S64).

[0084] Based on $A^{-1}(m, r)$, the inverter 503 recovers $w = s \| t = s_0 \| s_1 \| t$ (step S65). Subsequently, the OAEP + inverter 504 computes the exclusive OR of H(s) and each bit of t to recover R. Also, the OAEP + inverter 504 computes the exclusive OR of G(R) and each bit of s_0 to recover M (step S66).

[0085] Thereafter, the OAEP + inverter 504 verifies the validity of the padding according to whether $s_1 = H'(R \| M)$ is satisfied or not (step S67). If the padding is valid, the OAEP + inverter 504 decrypts the shared key encrypted ciphertext with the shared key encryption key M to output the plaintext X (step S68). Otherwise, the OAEP + inverter 504 rejects the ciphertext e as invalid and outputs $\perp$ (step S69).

BRIEF DESCRIPTION OF THE DRAWINGS

[0086] Fig. 1 (A) is a conceptual block diagram showing an encryptor adopting a padding application method according to the present invention; Fig. 1 (B) is a conceptual block diagram showing a conventional encryptor adopting an OAEP-based padding scheme.

Fig. 2 is a block diagram showing an example of a cryptographic communication system including an encryptor/ decryptor according to the present invention.

Fig. 3 is a functional block diagram showing the construction of an encryptor/ decryptor according to the first embodiment of the present invention.

Fig. 4 is a flowchart showing the key creation process according

to the first embodiment.

Fig. 5 is a flowchart showing the encryption process according to the first embodiment.

Fig. 6 is a flowchart showing the decryption process according to the first embodiment.

Fig. 7 is a functional block diagram showing the construction of an encryptor/ decryptor according to the second embodiment of the present invention.

Fig. 8 is a flowchart showing the key creation process according to the second embodiment.

Fig. 9 is a flowchart showing the encryption process according to the second embodiment.

Fig. 10 is a flowchart showing the decryption process according to the second embodiment.

DESCRIPTION OF CODES

[0087]

100	Encryptor
101	Plaintext input unit
102	OAEP + converter
103	Conversion function A-based converter
104	NTRU encryption section
105	Ciphertext output unit
200	Decryptor
201	Ciphertext input unit
202	NTRU decryption section
203	Random number recovery section
204	Conversion function A-based inverter
205	OAEP + inverter
206	Plaintext output unit
300	Key generator

301 Public information storage

302 Private key storage